In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

In [2]:

```
titanic_train = pd.read_csv("train.csv", index_col = 0)
titanic_test = pd.read_csv("test.csv", index_col = 0)
titanic_train.head()
```

Out[2]:

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
PassengerId											
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Analysis of the train dataset

In [3]:

```
titanic_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 891 entries, 1 to 891
Data columns (total 11 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Survived  891 non-null    int64  
 1   Pclass    891 non-null    int64  
 2   Name      891 non-null    object 
 3   Sex       891 non-null    object 
 4   Age       714 non-null    float64
 5   SibSp     891 non-null    int64  
 6   Parch     891 non-null    int64  
 7   Ticket    891 non-null    object 
 8   Fare      891 non-null    float64
 9   Cabin     204 non-null    object 
 10  Embarked  889 non-null    object 
dtypes: float64(2), int64(4), object(5)
memory usage: 83.5+ KB
```

In [4]:

```
# Not all columns are needed for the analysis, drop the columns not needed.
column = ['Cabin', 'Ticket', 'Name']
titanic_train.drop(columns = column, inplace = True)
```

In [5]:

```
titanic_train.isna().sum()
```

Out[5]:

```
Survived      0
Pclass        0
Sex           0
Age         177
SibSp         0
Parch         0
Fare          0
Embarked      2
dtype: int64
```

In [6]:

```
#fill age column with median
titanic_train.Age.fillna(titanic_train.Age.median(), inplace = True)
```

In [7]:

```
titanic_train.isna().sum()
```

Out[7]:

```
Survived    0
```

```
Out[7]: Survived    0
        Pclass      0
        Sex         0
        Age         0
        SibSp       0
        Parch       0
        Fare        0
        Embarked    2
        dtype: int64
```

In [31]:

```
titanic_train.dropna(inplace = True)
```

In [9]:

```
titanic_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 889 entries, 1 to 891
Data columns (total 8 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Survived  889 non-null    int64  
 1   Pclass    889 non-null    int64  
 2   Sex       889 non-null    object 
 3   Age       889 non-null    float64
 4   SibSp     889 non-null    int64  
 5   Parch     889 non-null    int64  
 6   Fare      889 non-null    float64
 7   Embarked  889 non-null    object 
dtypes: float64(2), int64(4), object(2)
memory usage: 62.5+ KB
```

In [10]:

```
titanic_train.head()
```

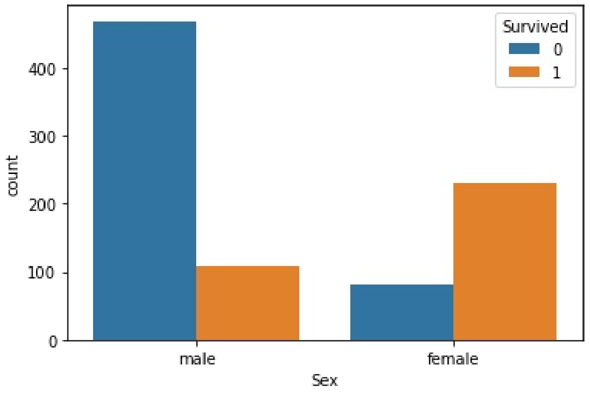
Out[10]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
PassengerId								
1	0	3	male	22.0	1	0	7.2500	S
2	1	1	female	38.0	1	0	71.2833	C
3	1	3	female	26.0	0	0	7.9250	S
4	1	1	female	35.0	1	0	53.1000	S
5	0	3	male	35.0	0	0	8.0500	S

In [11]:

```
#survivors based on sex
sns.countplot(data = titanic_train, x = 'Sex', hue = 'Survived')
```

Out[11]: <AxesSubplot:xlabel='Sex', ylabel='count'>



Convert the non numerical data to numerical data so as to predict the model

In [12]:

```
#Split data into features and label (X and y)
X = titanic_train.drop('Survived', axis = 1)
y = titanic_train['Survived']
```

In [13]:

```
# Transform the non numerical into numerical using sklearn
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer

cat_features = ['Sex', 'Embarked']
one_hot = OneHotEncoder()
transformer = ColumnTransformer([("one_hot", one_hot,
                                  cat_features)],
                               remainder = "passthrough")

transformed_X = transformer.fit_transform(X)
transformed_X
```

Out[13]:

```
array([[ 0.    ,  1.    ,  0.    , ...,  1.    ,  0.    ,  7.25  ],
       [ 1.    ,  0.    ,  1.    , ...,  1.    ,  0.    , 71.2833],
       [ 1.    ,  0.    ,  0.    , ...,  0.    ,  0.    ,  7.925 ],
       ...,
       [ 1.    ,  0.    ,  0.    , ...,  1.    ,  2.    , 23.45  ],
       [ 0.    ,  1.    ,  1.    , ...,  0.    ,  0.    , 30.    ],
       [ 0.    ,  1.    ,  0.    , ...,  0.    ,  0.    ,  7.75  ]])
```

In [14]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
np.random.seed(20)

#split data into train, test
X_train, X_test, y_train, y_test = train_test_split(transformed_X, y, test_size = 0.2)

clf = RandomForestClassifier(n_estimators = 100)
clf.fit(X_train, y_train)
```

Out[14]:

```
RandomForestClassifier()
```

In [15]:

```
clf.score(X_test, y_test)
```

Out[15]:

```
0.7752808988764045
```

In [16]:

```
y_preds = clf.predict(X_test)
```

In [17]:

```
pd.DataFrame({'Actual': y_test, 'Predicted': y_preds})
```

Out[17]:

	Actual	Predicted
PassengerId		
349	1	1
562	0	0
791	0	0
837	0	0
57	1	1
...	...	...
423	0	0
827	0	1
430	1	0
433	1	0
563	0	0

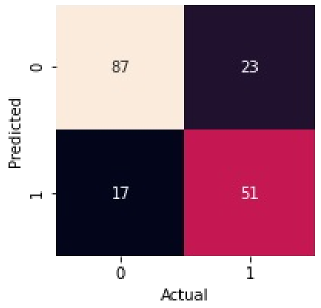
178 rows × 2 columns

In [18]:

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, precision_score, f1_score,
conf_mat = confusion_matrix(y_test, y_preds)

#Visualize confusion matrix using seaborn
def plot_mat(conf_mat):
    fig, ax = plt.subplots(figsize = (3,3))
    ax = sns.heatmap(conf_mat,
                     annot = True,
                     cbar = False)
    plt.xlabel("Actual")
    plt.ylabel("Predicted");

plot_mat(conf_mat)
```



In [19]:

```
# evaluate the classifier
print('Classifier metrics on titanic data set')

print(f'Accuracy: {accuracy_score(y_test, y_preds)*100:.2f}%')
print(f'Precision : {precision_score(y_test, y_preds)}')
print(f'Recall: {recall_score(y_test, y_preds)}')
print(f'F1: {f1_score(y_test, y_preds)}')
```

```
Classifier metrics on titanic data set
Accuracy: 77.53%
Precision : 0.6891891891891891
Recall: 0.75
F1: 0.7183098591549296
```

The model prediction score is 77.53%

Using the model built, predict the test data

In [20]:

```
titanic_test.head()
```

Out[20]:

	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
PassengerId										
892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

In [21]:

```
titanic_test.drop(columns =['Ticket', 'Cabin', 'Name'], axis = 1, inplace = True)
titanic_test.Age.fillna(titanic_test.Age.median(), inplace = True)
titanic_test.fillna(0,inplace = True)
```

In [22]:

```
titanic_test.head()
```

Out[22]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
PassengerId							
892	3	male	34.5	0	0	7.8292	Q
893	3	female	47.0	1	0	7.0000	S
894	2	male	62.0	0	0	9.6875	Q
895	3	male	27.0	0	0	8.6625	S
896	3	female	22.0	1	1	12.2875	S

In [23]:

```
# convert non numerical to numerical
cate_features = ['Sex', 'Embarked']
one_hot = OneHotEncoder()
transformer = ColumnTransformer([('one_hot',
                                  one_hot,
                                  cate_features)],
                                remainder='passthrough')
transformed_data = transformer.fit_transform(titanic_test)
```

In [24]:

```
predicted_test_data = (clf.predict(transformed_data))
```

In [25]:

```
gender = pd.read_csv('gender_submission.csv')
gender.head()
```

Out[25]:

	PassengerId	Survived
0	892	0
1	893	1
2	894	0
3	895	0
4	896	1

In [26]:

```
predicted_test_data
```

Out[26]:

```
array([0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1,
       1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1,
       1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1,
       1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1,
       1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1,
       0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0,
       1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1,
       0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0,
       1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0,
       0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1],
      dtype=int64)
```

In [29]:

```
#Load and save prediction
prediction = pd.DataFrame({'PassengerId': gender.PassengerId,'Survived': predicted_test_data})
prediction.to_csv('submission.csv', index = False)
print('Your prediction was successfully saved')
```

```
Your prediction was successfully saved
```

In [30]:

```
#check the saved prediction
test_pred = pd.read_csv('submission.csv', index_col = 0)
test_pred.head()
```

Out[30]:

	Survived
PassengerId	
892	0
893	0
894	0
895	1
896	1

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js